Gyanmanjari Innovative University

Syllabus
Gyanmanjari Diploma Engineering College
Semester-3 (Diploma)

**Subject:** Software Development Foundations – DET2CE13206

**Type of course:** Professional Core

**Prerequisite:** Basic computer literacy and familiarity with operating a keyboard, web browser, and file system. Students should have completed Semester 1 and 2 coursework. No prior programming experience is required.

**Rationale:** This course introduces students to the basic concepts of software foundation and software development practices. The course focuses on understanding software development processes, requirement collection, basic software design, simple UML diagrams, coding practices, documentation, and introductory testing concepts. Students will learn how software applications are planned, designed, documented, tested, and maintained using simple real-life examples and beginner-level mini projects.

**Teaching and Examination Scheme:**

| Teaching Scheme | | | Credits | Examination Marks | | Total Marks |
|---|---|---|---|---|---|---|
| CI | T | P | C | SEE | CCE | |
| 2 | 0 | 2 | 3 | 100 | 50 | 150 |

*Legends: CI-Class Room Instructions; T – Tutorial; P - Practical; C – Credit; SEE - Semester End Evaluation; CCE-Continuous and Comprehensive Evaluation.*

**Course Content:**

| Sr. No | Course Content | Hrs. | % Weightage |
|---|---|---|---|
| 1 | **Fundamentals of Software Engineering and Agile Development**<br><br>***Theory Topics:***<br><br>Software Concepts, Features and Classification of Software, Various Software Application Areas and Domains, Fundamentals of Software Engineering, Understanding the Software Process, Software Crisis and Common Software Myths, Software Quality Characteristics and Attributes, Overview of the Software Development Life Cycle (SDLC), Supporting Activities including Software Quality Assurance (SQA), Documentation, Project Planning and Management, Configuration Control, and Risk Handling, Software Development Process Models such as Waterfall Model, Incremental Model, Spiral Model, and Prototype Model, Concept of Agility and Cost of Change in Software Development, Agile Methodologies including Adaptive Software Development (ASD), Scrum, and Dynamic Systems Development Method (DSDM), Agile Project Collaboration and Management Tools such as JIRA and Trello.<br><br>***Practical:***<br><br>1. Creating a concept map to represent software characteristics and classifications by illustrating the relationships among system software, application software, and software development tools using Canva or PowerPoint.<br>2. Identifying and classifying various types of software by providing real-world examples and organizing them according to their purpose, functionality, and usage using MS Excel.<br>3. Preparing a short case study document describing the software crisis and explaining how software engineering principles and practices evolved to overcome related challenges using MS Word.<br>4. Analyzing a real-world software failure case to examine the contributing factors and identify common software myths that negatively influenced the project using MS Word. | 12 | 20% |

| | | | |
|---|---|---|---|
| | 5. Designing a visual diagram of the SDLC phases to clearly demonstrate the step-by-step stages of software development, from requirement analysis to maintenance, using tools such as Lucidchart, Draw.io (diagrams.net), Canva, Microsoft PowerPoint, Miro, and similar platforms.<br>6. Developing a paper-based prototype to visually represent the user interface and workflow of a simple application, allowing early-stage feedback and design improvements using Canva Whiteboard or traditional hand-drawn paper sketches.<br>7. Preparing a structured comparison table of major software process models to clearly present their phases, application scenarios, advantages, and limitations using Microsoft Excel or Microsoft Word.<br>8. Creating a simulated Agile Scrum workflow with a task board to monitor and display task movement through the To-Do, In Progress, and Done stages using Jira or Microsoft Excel.<br>9. Designing a burndown chart for a hypothetical Scrum sprint using Microsoft Excel to monitor and visualize the remaining work throughout the sprint duration.<br>10. Creating basic SQA and risk management tables to monitor quality assurance activities and identify possible project risks along with their impact and preventive actions using Microsoft Excel or Jira. | | |

**Examination Style:**

| Sr. No. | Evolution Method | SEE | CCE |
|---|---|---|---|
| 1 | **Software Process Model Selection for a Personal Project** Students will select a small software application idea, choose a suitable software process model, and justify their selection based on the project requirements, complexity, flexibility, and development needs. | - | 10 |
| 2 | **Software Development Risk Identification Activity** Students will identify possible technical, time, cost, resource, and requirement-related risks in a small software project and suggest suitable mitigation strategies in a structured table. | 10 | - |
| 3 | **Active Learning Activity (ALA): Software Process Evaluation and Prototype Modelling** Students will analyse a selected software application, illustrate its SDLC stages, compare suitable software process models, and prepare a basic prototype or workflow using appropriate digital tools. The completed work must be compiled and submitted as a single PDF. | 10 | - |
| | Total | 20 | 10 |

*Examination Style:*

1. **Software Process Model Selection for a Personal (10 Marks):**
   - Students will select a small application idea and prepare a brief explanation describing the most suitable software process model for its development, along with justification. The completed work should be saved as a PDF and uploaded to with detail.

| | | | |
|---|---|---|---|
| | 2. **Software Development Risk Identification Activity (10 Marks):**<br>• Students will imagine developing a small software project and identify possible risks related to: Technical challenges Time management Cost/resources User requirements. Students will also propose basic risk mitigation strategies and present them in a table format.<br><br>3. **Active Learning Activity (ALA): Software Process Evaluation and Prototype Modelling: (10 Marks):**<br>• Students will examine a selected software application using tools such as MS Word, PowerPoint, Draw.io, or Canva to categorize the software, illustrate SDLC stages, compare different software process models, and design a basic module prototype or workflow. Inclusion of Software Quality Assurance (SQA) and risk analysis tables is optional. All components must be compiled and submitted as a single PDF document. | | |
| 2 | **Requirements Engineering and System Specification**<br><br>***Theory Topics:***<br><br>Introduction to Requirement Engineering and the Requirement Engineering Process, including techniques for requirements gathering such as studying existing documents, interviews, questionnaires, record reviews, task analysis, and observation methods. It also covers tools used for documenting procedures and decisions, including decision concepts, decision trees, and decision tables. The topic further explains requirements analysis, preparation of the Software Requirements Specification (SRS) document, its format, and Institute of Electrical and Electronics Engineers standards for SRS documentation. Additional areas include SRS validation, components and characteristics of an SRS, different types of requirements such as functional and non-functional requirements, design and implementation constraints, required external interfaces, and other non-functional requirements. The syllabus also includes requirements validation and verification along with the use of Entity-Relationship (ER) diagrams. | 12 | 20% |

| | | | |
|---|---|---|---|
| | ***Practical:***<br><br>11. Designing a visual flow diagram of the Requirements Engineering process to illustrate how requirements are gathered, analyzed, documented, validated, and managed throughout the software development lifecycle.<br>12. Developing a requirement gathering report for a Student Management System using the interview method by documenting stakeholder questions, responses, and collected system requirements.<br>13. Creating a requirement gathering report for a Student Management System using the questionnaire technique, including prepared questions and analysis of the collected responses.<br>14. Preparing a requirement gathering report for a Student Management System using the observation method by recording user activities, workflows, and system interactions.<br>15. Developing a requirement gathering report for a Student Management System using the record review method through the analysis of existing documents, forms, and reports related to the system.<br>16. Creating a requirement gathering report for a Student Management System using task analysis by identifying and breaking down user tasks, activities, and workflows.<br>17. Developing a questionnaire containing 10–15 questions to gather requirements for a selected system or application.<br>18. Conducting a sample interview and preparing an interview summary that highlights the major user requirements and important observations.<br>19. Gathering real user requirements through a structured online survey to understand user needs, preferences, and expectations for a project.<br>20. Designing a decision tree for a selected decision-making scenario within the system. | | |

**Examination Style:**

| Sr. No. | Evolution Method | SEE | CCE |
|---|---|---|---|
| 1 | **Requirement Gathering Report for a Selected System** Apply suitable requirement-gathering techniques and prepare a structured report of the identified system requirements. | 10 | - |
| 2 | **Functional and Non-Functional Requirement Classification Activity** Identify and classify functional, non-functional, external interface, and implementation requirements for a selected system. | 10 | - |
| 3 | **Active Learning Activity (ALA): Requirement Engineering and Basic SRS Documentation Activity** Gather, analyse, classify, and document system requirements through a basic SRS and a suitable supporting diagram or decision tool. | - | 10 |
| | Total | 20 | 10 |

*Examination Style:*

1. **Requirement Gathering Report for a Selected System (10 Marks):**
   - Students will be given a sample project folder and must perform basic Git operation such as creating a repositories, adding files, committing changes, creating branches, and uploading files to github. Students must submit screenshots of repository structure and commit history along with a short explanation of the steps performed. This task evaluates practical understanding of Git fundamentals and repository management.

2. **Functional and Non-Functional Requirement Classification Activity (10 Marks):**
   - Students will imagine or select a small software project such as Student Management System, Library

| | | | |
|---|---|---|---|
| | Management System, Online Shopping System, Attendance Management System, Hospital Appointment System, or any similar application. Students will prepare a structured requirement classification table by identifying functional requirements, non-functional requirements, external interface requirements, and design or implementation constraints. Students will also write a brief explanation describing how proper requirement classification helps in preparing a clear and complete Software Requirements Specification (SRS) document.<br><br>3. **Active Learning Activity (ALA): Requirement Engineering and Basic SRS Documentation Activity (10 Marks):**<br>• Students will select a small software system or application such as Student Management System, Library Management System, Attendance Management System, Canteen Management System, Online Appointment System, College Event Registration System, or any similar application. Using tools such as MS Word, PowerPoint, Google Forms, MS Excel, Draw.io, or Canva, students will gather requirements using any two techniques such as interview, questionnaire, observation, record review, or task analysis. Students will categorize the collected requirements into functional requirements, non-functional requirements, external interface requirements, and design or implementation constraints. Students will also prepare a basic Software Requirements Specification (SRS) outline and include one supporting diagram or decision tool such as an ER diagram, decision tree, or decision table related to the selected system. All components must be compiled and submitted as a single PDF document. | | |
| 3 | **Software Design, Architecture and UML Modelling**<br><br>***Theory Topics:***<br><br>Introduction to system and software design concepts, including architectural design, coupling and cohesion, and different software design approaches such as function-oriented design and object-oriented design. The topic also covers various architectural styles including layered architecture, MVC, client-server architecture, and microservices. Additional areas include component-level design, interface design principles, secure design principles, and | 12 | 20% |

| | | | |
|---|---|---|---|
| | design strategies for scalability and reliability. It further explains UML diagrams such as object diagrams, class diagrams, methods, class relationships, use case diagrams, activity diagrams, state chart diagrams, and sequence diagrams. Practical modeling and diagram creation can be performed using tools like Draw.io (diagrams.net), Microsoft Visio, and Visual Paradigm Online.<br><br>***Practical:***<br><br>21. Demonstrating key software design concepts such as abstraction, modularity, cohesion, and coupling through simple visual examples to explain effective software design practices.<br>22. Creating diagrams of major software architectural styles, including Layered, MVC, Client-Server, and Microservices architectures, to understand system structure and communication flow visually.<br>23. Developing a component-level block diagram to represent software modules along with their input and output interfaces for better structural understanding.<br>24. Designing simple user interface screens by applying core UI principles such as consistency, simplicity, visibility, feedback, and error prevention.<br>25. Applying essential secure design principles using a structured checklist to ensure secure input handling, controlled access, and safe system behavior.<br>26. Preparing a basic scalability and reliability plan to ensure the software can support future growth while maintaining stable performance and availability.<br>27. Designing a Class Diagram for a Student Management System to visually represent classes, attributes, methods, and relationships within the system.<br>28. Creating a Sequence Diagram for a Student Management System to illustrate interactions between objects for a specific use case or scenario.<br>29. Developing an Activity Diagram for a Student Management System to model the workflow and sequence of activities involved in a particular process.<br>30. Creating a Use Case Diagram for a Student Management System to represent system functionalities and user interactions visually. | | |

**Examination Style:**

| Sr. No. | Evolution Method | SEE | CCE |
|---|---|---|---|
| 1 | **Software Architecture and Component Design Activity** Select a suitable architectural style and prepare architecture and component-level diagrams showing system modules, interfaces, and communication flow. | 10 | - |
| 2 | **Software Requirement Analysis and Prototype Design for a Selected System** Analyse system requirements, roles, modules, inputs, outputs, and constraints, and develop suitable interface wireframes or a basic prototype. | 10 | - |
| 3 | **Active Learning Activity (ALA): Software Architecture and UML Design Modelling Activity** Design the architecture and UML models of a selected system while considering modularity, security, scalability, reliability, and interface design. | - | 10 |
| | Total | 20 | 10 |

*Examination Style:*

1. **Software Architecture and Component Design Activity (10 Marks) :**
   - Students will select or imagine a small software system such as Student Management System, Library Management System, Attendance Management System, Online Food Ordering System, Hospital Appointment System, or any similar application. Students will identify a suitable software architectural style such as Layered Architecture, MVC Architecture, Client-Server Architecture, or Microservices Architecture. Students will prepare an architecture diagram and component-level block diagram showing major modules, input/output interfaces, and communication flow. Students will also

| | | | |
|---|---|---|---|
| | briefly explain how the selected architecture supports system organization, scalability, reliability, and maintainability.<br><br>**2. Software Requirement Analysis and Prototype Design for a Selected System (10 Marks) :**<br>• Students will select or conceptualize a small software system such as a Student Management System, Library Management System, Attendance Management System, Online Shopping System, or Canteen Management System. They will identify the system's functional and non-functional requirements, user roles, inputs, outputs, constraints, and major modules. Students will then prepare basic user-interface wireframes or a clickable prototype for the selected system using tools such as Figma, Canva, or Draw.io. A brief report explaining the identified requirements, system workflow, module descriptions, and prototype screens shall also be submitted.<br><br>**3. Active Learning Activity (ALA) : Software Architecture and UML Design Modelling Activity (10 Marks) :**<br>• Students will select a small software system or application such as Student Management System, Library Management System, Attendance Management System, Online Shopping System, Hospital Appointment System, Canteen Management System, or any similar application. Using tools such as Draw.io, Microsoft Visio, Visual Paradigm Online, Canva, or PowerPoint, students will design the software from a structural and behavioural perspective. Students will identify suitable design concepts such as abstraction, modularity, cohesion, and coupling, select an appropriate architectural style such as Layered, MVC, Client-Server, or Microservices, and prepare basic UML diagrams including class diagram, use case diagram, activity diagram, and sequence diagram. Students will also include a short explanation of interface design, secure design, scalability, and reliability considerations for the selected system. All components must be compiled and submitted as a single PDF document. | | |

| | | | |
|---|---|---|---|
| 4 | **Object-Oriented Analysis, Code Quality and Software Testing**<br><br>***Theory Topics:***<br><br>Introduction to object-oriented analysis and the GOAD (Game of Active Director) & the methodology, along with the applications of the software analysis and design process. The topic covers object-oriented design (OOD) goodness criteria, coding standards, and coding guidelines to ensure high-quality software development. It also includes concepts of code review, software documentation, and various software testing activities and strategies. Different testing techniques such as unit testing, black-box testing, white-box testing, debugging, integration testing, and system testing are also discussed. Additional topics include test case and test suite design, testing of conventional applications, object-oriented applications, web applications, and mobile applications. The syllabus further introduces testing tools such as WinRunner and LoadRunner, along with the preparation and management of bug reports.<br><br>***Practical:***<br><br>31. Applying coding standards and formatting practices to develop clean, organized, and easy-to-read code within a programming environment.<br>32. Reviewing a provided code sample to identify errors and suggest improvements, promoting best practices for readability, maintainability, and correctness.<br>33. Preparing clear software documentation that describes the program objective, input and output details, logic flow, and execution outcomes.<br>34. Designing well-structured test cases and arranging them into a test suite to systematically verify program functionality and expected behavior.<br>35. Using Selenium IDE to record, replay, and validate automated user interactions on a live web application.<br>36. Performing unit testing on a function that computes the total bill amount by including tax and discount calculations.<br>37. Designing black-box testing test cases using Equivalence Partitioning for validating a User Age input field.<br>38. Simulating mobile application testing through an online emulator to assess website responsiveness, layout, and usability across different mobile devices. | 12 | 20% |

39. Performing statement coverage testing using white-box testing techniques for a program that determines whether a number is even or odd.
40. Conducting Integration Testing on the Login and Dashboard modules to verify proper interaction between components.

**Examination Style:**

| Sr. No. | Evolution Method | SEE | CCE |
|---|---|---|---|
| 1 | **Code Review, Documentation, and Unit Testing Activity** Review and improve a given program by applying coding standards, preparing documentation, and designing suitable unit test cases. | 10 | - |
| 2 | **Test Case Design, Integration Testing, and Bug Report Activity** Design a complete test suite for connected software modules, perform integration testing, and prepare structured bug reports. | 10 | - |
| 3 | **Active Learning Activity (ALA): Software Quality, Code Review, and Testing Documentation Activity** Apply coding standards, code review, documentation, test-case design, selected testing techniques, and defect reporting for a software module. | - | 10 |
| | Total | 20 | 10 |

***Examination Style:***

1. **Code Review, Documentation, and Unit Testing Activity (10 Marks) :**
   - Students will be given or will select a small program such as total bill calculation, student result calculation, even-odd checking, simple login validation, or discount calculation. Students will review the code by checking coding standards, naming conventions, indentation, readability, logical correctness, and maintainability. Students will improve the code where required, prepare

| | | | |
|---|---|---|---|
| | short software documentation, and design unit test cases to verify the program functionality. This examination focuses on clean coding, code review ability, documentation quality, and basic unit testing skill.<br><br>**2. Test Case Design, Integration Testing, and Bug Report Activity (10 Marks) :**<br>• Students will select or imagine a small software system containing at least two connected modules such as Login and Dashboard, Registration and Login, Product Selection and Billing, Student Entry and Report Generation, or Appointment Booking and Confirmation. Students will design a test suite containing different types of test cases, including normal, invalid, boundary, and integration test cases. Students will also identify possible defects and prepare a structured bug report. This examination focuses on practical testing ability, test case coverage, integration testing, and defect reporting.<br><br>**3. Active Learning Activity (ALA) : Software Quality, Code Review, and Testing Documentation Activity (10 Marks):**<br>• Students will select or develop a small software module such as Login System, Billing System, Student Marks Calculation System, User Registration Form, Attendance Module, Shopping Cart Module, or any similar module. Using tools such as MS Word, MS Excel, Selenium IDE, online compiler, browser developer tools, mobile emulator, or any programming environment, students will apply coding standards, perform code review, prepare software documentation, design test cases, and prepare a basic bug report. Students will also include at least one testing approach such as unit testing, black-box testing, white-box testing, integration testing, mobile testing, or web application testing. All components must be compiled and submitted as a single PDF document. | | |

| | | | |
|---|---|---|---|
| 5 | **Software Project Management, Estimation and Risk Management**<br><br>***Theory Topics:***<br><br>Introduction to project management, including its definition, importance, and the characteristics of software projects. The topic covers the software project life cycle, the WI-IFI principle, and the roles and responsibilities of a project manager. It also explains project planning concepts such as Work Breakdown Structure (WBS), project scheduling, and cost estimation techniques including Line of Code (LOC) and Function Point analysis. Additional areas include the use of Gantt charts, PERT, and CPM techniques for project planning and tracking. The syllabus further discusses project risk management, including identifying risks, risk projection, risk refinement, risk categories, probability and impact analysis, risk prioritization, and strategies for risk mitigation, monitoring, and management. It also introduces the RMMM (Risk Mitigation, Monitoring, and Management) plan, verification and validation processes, and software quality standards such as International Organization for Standardization and CMMI Institute.<br><br>***Practical:***<br><br>41. Develop a hierarchical Work Breakdown Structure (WBS) for a Library Management System by dividing the project into modules, submodules, and individual tasks.<br>42. Create a Work Breakdown Structure (WBS) for an E-commerce Website project to illustrate task decomposition and project organization.<br>43. Prepare a project timeline for a Student Management System using a Gantt chart to schedule activities and assign responsibilities.<br>44. Design a Gantt chart for a software development project containing activities such as Requirements Analysis, Design, Coding, Testing, and Deployment with specified durations.<br>45. Construct a Gantt chart for a Mobile Application Development project with at least six project tasks and realistic timelines.<br>46. Create a Gantt chart for a Web Development Project to demonstrate parallel execution of activities such as front- | 12 | 20% |

end development, back-end development, database setup, and testing.

47. Given a software project schedule, prepare a Gantt chart and identify task dependencies, overlapping activities, start dates, and end dates.
48. Estimate software development effort and cost using the Lines of Code (LOC) technique and represent the calculations in a spreadsheet.
49. Calculate software size and development cost using the Function Point Analysis method in a structured spreadsheet format.
50. Track and monitor project progress using a Gantt chart by updating task completion percentages and project status over time.

**Examination Style:**

| Sr. No. | Evolution Method | SEE | CCE |
|---|---|---|---|
| 1 | **WBS, Gantt Chart, and Project Tracking Activity** Develop a WBS and Gantt chart showing project tasks, timelines, dependencies, responsibilities, progress, and critical activities. | 10 | - |
| 2 | **Software Cost Estimation and Risk Management Activity** Estimate software effort and cost using LOC or Function Point Analysis and prepare a prioritized risk management and RMMM plan. | 10 | - |
| 3 | **Active Learning Activity (ALA): Software Project Planning, Estimation, and Risk Management Activity** Prepare a complete software project plan covering WBS, scheduling, cost estimation, progress tracking, risk management, quality, verification, and validation. | - | 10 |
| | Total | 20 | 10 |

| | | | |
|---|---|---|---|
| | *Examination Style:*<br><br>**1. WBS, Gantt Chart, and Project Tracking Activity (10 Marks) :**<br>• Students will select or imagine a software project such as Student Management System, Library Management System, E-commerce Website, Mobile Application Development Project, Web Development Project, or any similar application. Students will prepare a Work Breakdown Structure (WBS) by dividing the project into modules, submodules, and individual tasks. Based on the WBS, students will prepare a Gantt chart containing task duration, start date, end date, dependencies, overlapping activities, and assigned responsibilities. Students will also update project progress using completion percentage and identify delayed or critical tasks. This examination focuses on project planning, scheduling, dependency analysis, and progress monitoring.<br><br>**2. Software Cost Estimation and Risk Management Activity (10 Marks) :**<br>• Students will select or imagine a software project such as Library Management System, Online Shopping System, Attendance Management System, Hospital Appointment System, Online Examination System, or any similar application. Students will estimate the software development effort and cost using either the Lines of Code (LOC) technique or Function Point Analysis. Students will prepare the calculation in a structured table. Students will also identify project risks, classify them into risk categories, analyze their probability and impact, prioritize them, and prepare a basic RMMM plan for the most important risks. This examination focuses on estimation, analytical thinking, risk prioritization, and project control.<br><br>**3. Active Learning Activity (ALA): Software Project Planning, Estimation, and Risk Management Activity (10 Marks) :**<br>• Students will select a medium-level software project such as Library Management System, E-commerce Website, Student Management System, Hospital Appointment System, Mobile Food Ordering Application, Online Examination System, or any similar software project. | | |

| | | | |
|---|---|---|---|
| | Using tools such as MS Excel, MS Word, PowerPoint, Canva, Draw.io, or project management tools, students will prepare a complete project planning document. The document must include a hierarchical Work Breakdown Structure (WBS), project schedule using a Gantt chart, task dependencies, effort and cost estimation using LOC or Function Point Analysis, project progress tracking, and a structured risk management plan. Students will also prepare a basic RMMM plan for major risks and briefly connect the project with verification, validation, and software quality standards such as ISO or CMMI. All components must be compiled and submitted as a single PDF document. | | |

**Suggested Specification:**

| Distribution of Theory Marks (Revised Bloom's Taxonomy) | | | | | | |
|---|---|---|---|---|---|---|
| **Level** | **Remembrance (R)** | **Understanding (U)** | **Application (A)** | **Analyze (N)** | **Evaluate (E)** | **Create (C)** |
| Weightage % | 10% | 15% | 30% | 15% | 10% | 20% |

*Note: This specification table shall be treated as a general guideline for students and teachers. The actual distribution of marks in the question paper may vary slightly from above table.*

## Course Outcome:

| After learning the course the students should be able to: | |
|---|---|
| CO1 | Explain fundamental software concepts, processes, quality attributes, and project management practices. |
| CO2 | Analyse and document software requirements using SRS, ER diagrams, and standard techniques. |
| CO3 | Design software systems using design principles, UML diagrams, and modelling tools. |
| CO4 | Apply object-oriented design, coding standards, and testing techniques to ensure software quality. |
| CO5 | Plan and manage software projects considering estimation, scheduling, risks, quality, and standards. |

**Instructional Method:**

The course delivery method will depend upon the requirement of content and needs of students. The teacher, in addition to conventional teaching methods by black board, may also use any tools such as demonstration, role play, Quiz, brainstorming, MOOCs etc.

From the content 10% topics are suggested for flipped mode instruction.

Students will use supplementary resources such as online videos, NPTEL/SWAYAM videos, e-courses, Virtual Laboratory.

The internal evaluation will be done on the basis of the Active Learning Assignment.

Practical/Viva examination will be conducted at the end of semester for evaluation of performance of students in the laboratory.

**Reference Books:**

[1]Software Engineering — *Pearson Education.*
[2] Software Engineering: *A Practitioner's Approach — McGraw Hill Education.*
[3] HTML and CSS: *Design and Build Websites — Wiley Publishing.*
[4] *JavaScript and jQuery: Interactive Front-End Web Development — Wiley Publishing.*
[5] Pro Git — *Git SCM Documentation.*

## Suggested Rubrics:

Suggested Assessment Guidelines

**Module-1: Fundamentals of Software Engineering and Agile Development**

| **Software Process Model Selection for a Personal Project (10 Marks)** | | |
|---|---|---|
| **Criteria** | **Description** | **Marks** |
| Project Analysis and Process Model Selection | Clear understanding of the selected software project, identification of its requirements and development needs, and selection of an appropriate software process model. | 05 |
| Process Model Evaluation and Justification | Logical justification of the selected process model based on project complexity, flexibility, requirement stability, risks, advantages, and limitations. | 05 |
| | **Total** | 10 |

| **Software Development Risk Identification Activity (10 Marks)** | | |
|---|---|---|
| **Criteria** | **Description** | **Marks** |
| Risk Identification and Analysis | Identification and classification of relevant technical, schedule, cost, resource, and requirement-related risks, including their probability and impact. | 05 |
| Risk Mitigation and Management | Preparation of suitable mitigation, monitoring, and management strategies for controlling the identified software project risks. | 05 |
| | Total | 10 |

**Module-2: Requirements Engineering and System Specification**

| **Requirement Gathering Report for a Selected System (10 Marks)** | | |
|---|---|---|
| **Criteria** | **Description** | **Marks** |
| Requirement-Gathering Technique and Data Collection | Appropriate selection and application of requirement-gathering techniques such as interviews, questionnaires, etc. or task analysis to collect relevant system information. | 05 |
| Requirement Analysis and Report Preparation | Clear analysis, organisation, and documentation of the gathered requirements in a structured report covering system objectives, stakeholders, and major system requirements. | 05 |
| | **Total** | 10 |

| Functional and Non-Functional Requirement Classification Activity (10 Marks) | | |
|---|---|---|
| **Criteria** | **Description** | **Marks** |
| Requirement Identification and Classification | Accurate identification and classification of functional requirements, and design or implementation constraints for the selected system. | 05 |
| Requirement Clarity and Justification | Clear, complete, and logically organised presentation of the classified requirements with suitable explanations and justification of their role in preparing an effective Software Requirements Specification document. | 05 |
| | Total | 10 |

**Module-3: Software Design, Architecture and UML Modelling**

| Requirement Gathering Report for a Selected System (10 Marks) | | |
|---|---|---|
| **Criteria** | **Description** | **Marks** |
| Architecture Selection and Design | Appropriate selection of a software architectural style and clear preparation of an architecture diagram showing major system modules and overall structure. | 05 |
| Component and Interface Modelling | Accurate representation of components, input and output interfaces, dependencies, and communication flow with suitable explanation of scalability, reliability, and maintainability. | 05 |
| | **Total** | 10 |

| Software Requirement Analysis and Prototype Design for a Selected System (10 Marks) | | |
|---|---|---|
| **Criteria** | **Description** | **Marks** |
| Requirement and System Analysis | Clear identification and analysis of functional and non-functional requirements, user roles, inputs, outputs, constraints, workflows, and major system modules. | 05 |
| Prototype Design and Documentation | Development of clear and suitable wireframes or a basic prototype with consistent interface design, logical navigation, and appropriate supporting documentation. | 05 |
| | Total | 10 |

**Module-4: Object-Oriented Analysis, Code Quality and Software Testing**

| Code Review, Documentation, and Unit Testing Activity (10 Marks) | | |
|---|---|---|
| **Criteria** | **Description** | **Marks** |
| Code Review and Improvement | Identify coding errors and improve the program by applying suitable coding standards, naming conventions, indentation, readability, logical correctness, and maintainability practices. | 05 |
| Documentation and Unit Testing | Prepare clear software documentation and design suitable unit test cases covering valid, invalid, and boundary conditions with expected results. | 05 |
| | **Total** | 10 |

| Test Case Design, Integration Testing, and Bug Report Activity (10 Marks) | | |
|---|---|---|
| **Criteria** | **Description** | **Marks** |
| Test Suite Design and Integration Testing | Prepare a structured test suite containing normal, invalid, boundary, and integration test cases, and verify the interaction between connected software modules. | 05 |
| Defect Identification and Bug Reporting | Identify software defects and prepare a complete bug report containing defect description, reproduction steps, expected result, actual result, severity, and status. | 05 |
| | Total | 10 |

**Module-5 Software Project Management, Estimation and Risk Management**

| WBS, Gantt Chart, and Project Tracking Activity (10 Marks) | | |
|---|---|---|
| **Criteria** | **Description** | **Marks** |
| Work Breakdown Structure and Project Scheduling | Prepare a clear hierarchical WBS and a suitable Gantt chart showing project tasks, subtasks, timelines, dependencies, overlapping activities, responsibilities, and milestones. | 05 |
| Project Tracking and Critical Activity Analysis | Update project progress using completion percentages, identify delayed or critical activities, and analyse the effect of task dependencies on project completion. | 05 |
| | **Total** | 10 |

| Software Cost Estimation and Risk Management Activity (10 Marks) | | |
|---|---|---|
| **Criteria** | **Description** | **Marks** |
| Software Effort and Cost Estimation | Estimate software size, development effort, time, and cost using either the Lines of Code method or Function Point Analysis with clear and accurate calculations. | 05 |
| Risk Analysis and RMMM Planning | Identify, classify, prioritise, and analyse project risks based on probability and impact, and prepare suitable risk mitigation, monitoring, and management strategies. | 05 |
| | Total | 10 |